Automated Testing document

Test Report

Thomas van der Molen

IPS3-DB03

|  |  |
| --- | --- |
| **Project Information** | |
| Project members | Thomas van der Molen (4168003) |
| Project name | Text Adventure |
| Version | 1.0 |

Table of Contents

[Version History 2](#_Toc93321511)

[Introduction 3](#_Toc93321512)

[Front-end 3](#_Toc93321513)

[Back-end 4](#_Toc93321514)

[Entity Manager 4](#_Toc93321515)

[Game Manager 4](#_Toc93321516)

# Version History

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Change** |
| 1.0 | 17-01-2022 | Created File |

# Introduction

The Text Adventure project is split into 3 services: The front-end service, the back-end entity manager, and the back-end game manager. All services have been tested on their most important functionalities / interactions.

# Front-end

**27 tests**

For my front-end React App, I have decided to use Cypress to test my logic, I have also decided to write my tests as integration tests with no dependencies on other running services (also because my GP frontend tests are exclusively end-to-end tests).

Because I test all my services and I think having tests rely on as little outside dependencies as possible, I have decided to test my front-end in this structure, which does mean that any interaction like logging in will have to be mocked.

For my testing coverage I have made sure to cover all routes and the basic interactions on set routes, this covers the homepage and game page and any user input possible on set pages, such as logging in, drawing, creating an adventurer, etc.

Furthermore, I have connected my CI/CD to with [cypress.io](https://www.cypress.io/) so that all ran tests will be documented on there, including videos and images of the tests.

# Back-end

For my back-end I have used Xunit to write my tests, in combination with Moq and Shouldly. For my testing I have decided to use both unit tests and integration tests. I have primarily focused on writing integration tests because these cover more code while taking less time.

## Entity Manager

**68 tests**

For my coverage I have tested all the API endpoints, this covers most of the project because the entity manager’s main job is handling persistent data together with its connected database.

Graphical user interface, text

Description automatically generatedWith the relatively large number of tests written I have a code coverage % of 51,90%. This covers 97,44% of the database context, 100% of the controller endpoints accessed directly by users (this excludes endpoints connecting the entity manager and game manager services) and 98,50% of the services.

|  |  |
| --- | --- |
| **Test** | **Duration** |
| Class: WeaponControllerTests Passed (11) | 607 ms |
| Class: UserControllerTests Passed (13) | 1,9 sec |
| Class: TestDbContextFactoryTests Passed (2) | 540 ms |
| Class: RoomControllerTests Passed (15) | 719 ms |
| Class: HubAdventurerControllerTests Passed (11) | 609 ms |
| Class: EnemyServiceTests Passed (2) | 5 ms |
| Class: DrawingControllerTests Passed (4) | 550 ms |
| Class: AdventurerServiceTests Passed (10) | 683 ms |

## Game Manager

**18 tests**

For my coverage I have tested the main command flow for a user exploring the world (this means an adventurer in the exploring state), I have chosen to test this because I have spent a relatively large amount of time on previous tests and by testing these interactions, I will have covered the main flow and logic of my game, including handling a connected user’s data and handling a command sent from a User.

Graphical user interface, text

Description automatically generatedTo test this project, I have had to mock the SignalR connected created when a user resumes a game, this took me quite a bit of time and has taught me a lot about how Moq and mocking in general works. With the tests written I cover 37.72% of the project, with 67% of the session logic, 74,19% of the command service and 49% of the commands.

|  |  |
| --- | --- |
| **Test** | **Duration** |
| Class: SessionManagerTests Passed (3) | 4 ms |
| Class: CommandServiceTests Passed (15) | 103 ms |